HOCHSCHULE TRIER
Trier University of Applied Sciences
Informatik - Computer Science

# Mobile Anwendung für die Kostenschätzung mit Android

## Mobile Application for Cost Estimations in Android

Oliver Fries

Bachelor-Abschlussarbeit

Betreuer: Prof. Dr. Georg Rock

Trier, 29.02.2016

# Abstract

Die Wichtigkeit von Kostenschätzungen in IT-Projekten steigt durch immer komplexere und größere Projekte stetig an. Schwierigkeiten bei der Kostenschätzung liegen im Zugriff auf bereits vorhandene Schätzungen, die zur Verbesserung der Schätzwerte beitragen, sowie der Auswahl relevanter Projekte, damit nicht alle verfügbaren betrachtet werden müssen. Diese Arbeit zielt darauf ab, den Prozess der Kostenschätzung als mobile Anwendung umzusetzen und die genannten Probleme mit einem dynamischen Prozess durchführen zu können. Hierbei steht die Umsetzung des Function Point Verfahrens im Fokus, sowie die Möglichkeit mit einem Vergleich der Projekte auf vorangegangene Schätzungen zuzugreifen. Diese mobile Anwendung wurde als Android Applikation umgesetzt und trägt den Namen MobileEstimate. Es ergeben sich dadurch neue Möglichkeiten für Projektmanager und Projektteams zur schnellen und einfachen Kostenanalyse bei IT Projekten und zum Monitoring der Projektkosten.

The importance of cost estimation in IT projects is constantly increasing through more complex and larger projects. Some of the difficulties in cost estimations are the access to existing estimations, which help to improve the estimation, and the selection of relevant projects in order to not consider all available projects. This paper aims to implement the cost estimation as a mobile application with a dynamic process to approach the specified problems. Implementation of the Function Point method is the focus of this paper and the possibility to compare projects and get access to their cost estimation. The mobile application was implemented as an Android application and is called MobileEstimate. This results in new opportunities for project managers and project teams for fast and easy cost analysis for IT projects and the monitoring of project costs.

# Contents


**1 Introduction** . . . . . 1

**2 Theoretical Background** . . . . . 3
2.1 Cost estimation in software engineering . . . . . 3
2.2 Estimation Techniques . . . . . 7
2.3 Software for Cost Estimations - State of the art . . . . . 15

**3 Application concept** . . . . . 19
3.1 Project planning . . . . . 19
3.2 Architecture . . . . . 23
3.3 Components . . . . . 23
3.4 Database design . . . . . 33
3.5 User Interface . . . . . 34
3.6 Adjusted Estimation Process . . . . . 34

**4 Implementation** . . . . . 35
4.1 Project Structure . . . . . 35
4.2 Use of an exisiting Database in Android . . . . . 35
4.3 The Database Helper . . . . . 35
4.4 Using multilingual Strings with the Databse and Resource Files . . . . 35
4.5 Calculate Person Days . . . . . 35
4.6 Find related Projects . . . . . 35
4.7 ListView Elements and ViewHolder . . . . . 35

**5 Software Test** . . . . . 36
5.1 Test Cases . . . . . 36
5.2 User Review . . . . . 36

**6 Conclusion** . . . . . 37
6.1 Results . . . . . 37
6.2 Retrospection . . . . . 37
6.3 Outlook . . . . . 37

**References** . . . . . 38

**Erklärung der Kandidatin / des Kandidaten** . . . . . 41

# 1

# Introduction

Most of the contracts IT companies subscribe are projects and these are notorious for going past their deadline and over their budget. According to the study of Capgemini in 2014 [DU14], the importance of cost estimation increases every year. The study asked for the most important requirements in the IT for the next years, with the top requirement to increase the efficiency, which means to lower the costs and to meet determined deadlines. This will enhance the effort companies have to take in planning their projects.
All businesses want to lower the risk of delayed or canceled projects. This results in more effort IT companies have to take in requirements engineering and cost estimation to give their clients an accurate estimation of the upcoming project. As a result of the increase in requirements engineering and, subsequently, the cost estimation will lower the budget overrun. 6% to 12% of the project time will lower the risk of a cost overrun. The budget overrun is limited with this to a maximum of 50% of the estimated cost[PA10]. Which means, that a higher effort in requirements engineering and cost estimation will lower the chances of failed projects.
Therefore cost estimation is an important element for planning software projects and can be responsible for successful or failed projects. It is even more important to estimate as precisely as possible to guide the project to success. There are several methods for these estimations that can be used at different phases of the project. These methods of estimation lean their result on the information they get from the development process and the artifacts of the particular project phase. These include requirement documents, diagrams or the program code itself. All available artifacts are depending on the used process model and the project phase [HU11]. Based on the described information, the actual project can be categorized, to find the 'best fitting' estimation method for the current estimation. These methods can be time-consuming and related projects can often only be found in the own company context or are based on experiences.
This paper aims to develop a mobile application which supports the Function Point estimation. The application aims to makes the estimation process in IT-projects simpler and more efficient. To achieve a better way for estimating costs the most important design guideline was 'Only show what I need when I need it' [GO16]. The comparison between projects has to be formalized and implemented. The idea is to give the user an overview over terminated projects and how they were esti-

mated. The user has the possibility to transfer such an estimation to a new project or get a quick view how much man days it took.
The developed application MobileEstimate should prove that cost estimation on mobile devices is possible. It should allow to estimate the costs of a project with function point method and among the existing projects related projects should be displayed. Estimation results from a related project are possible to be viewed in the application and transfered to another estimation.

# 2

# Theoretical Background

This chapter describes the fundamentals for the understanding of this paper. The cost estimation process in IT projects and the different methods to calculate the cost of a project are described in this chapter. The state of the art report combined with the market description will give a short overview over the situation about software estimation tools on the market and the possibility of a mobile solution of cost estimations. Android as the chosen platform and Java as the programming language will not be described in detail here and are assumed to be known.

## 2.1 Cost estimation in software engineering

The most expensive components of computer systems are software products. While private clients are mostly interested in the final price of a product, business clients of IT companies typically want to know the costs of the software before project launch. As analyzed in the *IT-Trends* study from Capgemini [DU14] the IT budget of companies is growing up to 10% every year. Whether developing a new project or standardizing existing software, the project costs are always on the main focus by the project management. As human resources is the biggest part of software costs, project managers and especially business clients want to know the estimated spendings and completion time of a project. Most of the estimation methods focus on this aspect and give the result in man days. These estimated days can then be converted into the real costs.
Basically the cost estimation in software engineering wants to answer following questions:

1. How much effort is required to complete the project?
2. How much days are needed to complete the project?
3. What is the total cost of the project?

While projects are a living thing, the effort may change due to unexpected difficulties. For a precise estimation of the total costs an adjustment cannot be avoided and it can be useful to change the estimation method in a later project phase. This means that the estimation process is not an one-time thing but will change through the life-time of a project.

### 2.1.1 Estimation Process

It is common to create the first cost estimation before the system design, but also for monitoring purposes, milestones or if the client wants an overview of the project. Each time an actual cost estimation is needed the estimation process is executed which is a set of techniques and procedures that are used to derive the software cost estimate. Kathleen Peters described the basic process of an estimation, as seen in figure 2.1, as it is common in the industry [KA16]. As can bee seen from figure 2.1, there are seven steps in the estimation process. The first part is to collect the initial requirements which is essential to know what the project is about and evaluate the approximate project size. With an selected estimation method, which are described in section 2.1.3, the evaluated size of the project is then estimated. Afterwards the effort in man-days is calculated from which the cost schedule is created. Data from older projects can be included into the cost estimation. In the process step to approve the estimation, it has to be decided, if the costs are acceptable or if range of functions has to be shortened and the re-estimation has to bee started. If the cost estimation is acceptable the development of the product can start or continue.

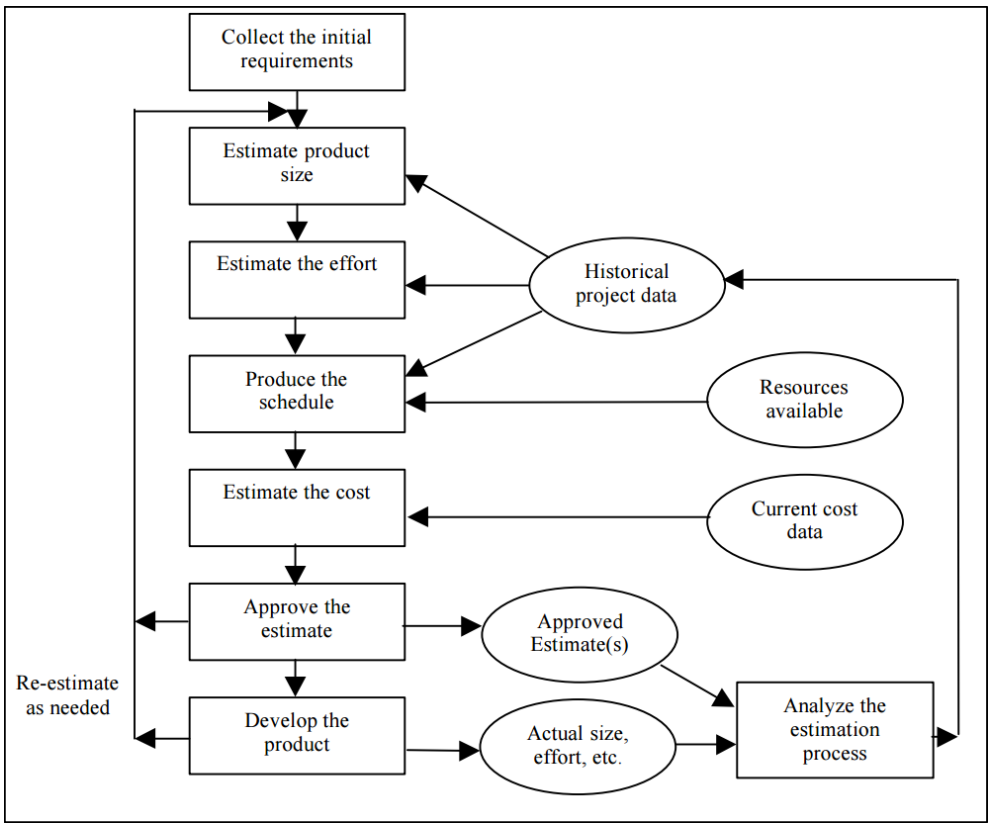


**Fig. 2.1.** - The Basic Project Estimation Process
Source: Peters, Kathleen - Software Project Estimation, Page 3

In this classical view of the estimation process there are four outputs generated which can be described as following:

1. Actual Size - the size of the project as a numerical value to make it comparable.
2. Manpower Loading - the amount of personnel that is allocated to the project.
3. Project Duration - the time that is needed to complete the project.
4. Effort - the amount of effort required to complete the project is usually measured in units as man-days (MD) or person-months (PM).

As described before, the estimation process can be triggered at any time in the project to re-estimate the costs. Depending on the project stage another estimation method than used before can be more precise.
The overview shown in fig. 2.2 shows that for example the SLIM method, is more suitable at the beginning of a project, whereas the ZKP method is more suitable after the system design stage. Most of the estimation methods can be used after the study stage. This is because a rough overview of the project size exists after the study stage. Different estimation methods may also change the evaluation output, which is one of the difficulties of cost estimations.

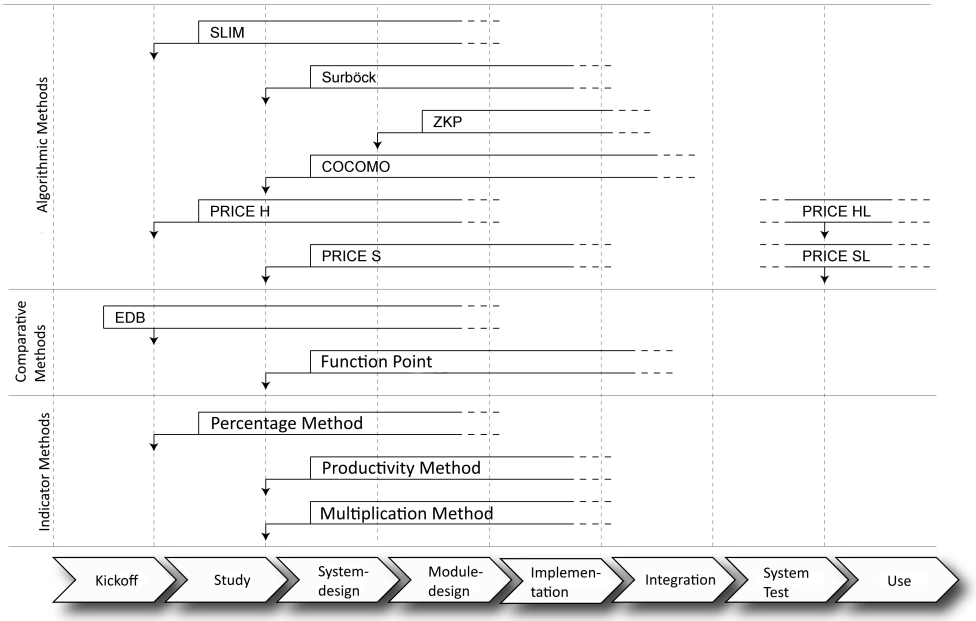


**Fig. 2.2.** - Starting Points of Estimation Methods
Source: `http://winfwiki.wi-fom.de/index.php/Methoden_und_Verfahren_der_Aufwandsch%C3%A4tzung_im_Vergleich`

### 2.1.2 Difficulty of estimations

One of the problems in estimating costs is that the actual code is only a small part of the project. Beside project planning there are many administrative tasks

to do, like coordination of the project or searching and fixing errors. Most estimation methods evaluate the estimated time for the implementation only and too little or no time is evaluated for the non implementation tasks. This resolves in an underestimation of the non implementation tasks or an overestimation if they are predicted as a high value [WI05].
Most project managers rely on their experience from past projects. This is an advantage, but as technology changes fast and new projects inherit new problems there is no prior experience for some parts of the project. That can make experiences from prior projects useless. Because of the unique nature of projects it is common that a new project has big parts where no experience exists. Another difficulty of estimations is the fact that people who are inherited in the project have a more positive outlook and mostly underestimate the costs. Also, customers often got a target time for the project, which leads in adjustment of the cost estimation. This leads to budget overrun if the needed resources are not available for the project [WI16].
It is also not guaranteed that the estimated costs are accurate and stay within the budget. An estimation can easily go past their estimated target as new technologies and unexpected difficulties are commonplace. A partial requirements engineering can also cause an inaccurate cost estimation due to unexpected difficulties in the implementation.

### 2.1.3 Methods for estimation

Estimation methods are different metrics to calculate the costs of a project and there are different approaches to categorize the estimation methods. The categorization used in this paper is based on the book "Management von IT-Projekten" by Hans W. Wieczorrek, where the methods are subdivided in algorithmic, comparative, operating figures and expert discussion [WI05].
All estimation techniques have in common, that only with a combined use of these different estimation methods a suitable result can be achieved as a measured value for the project. For the evaluation of the needed effort the underlying metric is used to calculate the effort for the project. On the basis of charge rates the effort size is calculated out of it.

#### Algorithmic Method

The algorithmic method uses a closed formula, which is based on empiric evaluation of already terminated projects or on existing mathematically models. Different forms of this method are the weight and the sampling method, which only differ in their usage.
The accuracy of the estimation depends primarily on the precision of the influence factors [WI05]. The algorithmic method always connects measurable project sizes, such as lines of code and implementable features, with influencing factors to get

the result, represented as required effort in personnel costs. The basic formula, as described by Wieczorrek [WI05]:

$$Personnelcosts = f(resultquantity, influencingfactors) \tag{2.1}$$

### Comparative Method

Not based on a formula or numerical connection, the comparative method tries to create a reference between the current project and past projects. Therefore projects from the own company or the same industry sector are analyzed with appropriate comparison methods. This estimation method has the advantage, that it can be used early in project development [WI05]. This method can be used for hardware and software projects.

### Key Figures Method

Estimation methods based on key figures can be differed to multiplier and percentage method. The multiplier method uses units of power as the base to estimate the total expenditure, whereas the percentage method uses the effort of a project stage to estimate the effort for the next stage.
After project completion a post calculation determines the total project costs and the amount of specific types of costs. In order to calculate these costs, they will be divided by the scope of the developed product. This results in new key figures which can be used for new projects by multiplication of the estimated scope with the appropriate key figures. Regular actualization of the key figures is necessary for right results [WI05].

### Expert Discussion Method

As a quantitative and heuristic method the expert discussion uses knowledge from selected groups of people. It differentiates between four kinds: single person interview, multiple person interview, Delphi method and assessment meeting.
The advantage of this estimation method is, that they are useful for all project types. But they have the risk of a strongly affection by subjective opinions and the experience of the interviewees. As a result, expert discussions should never be used for complete projects but only for sub projects [WI05].

## 2.2 Estimation Techniques

The existing estimation techniques rely on experience-based judgments by project managers who created, by combining estimation methods, techniques that can be used to estimate the effort of a project. Most of the estimation techniques became popular in the 80's. With the agile projects becoming more popular estimation

techniques for these project types are rising.
Because of the fundamental uniqueness nature of projects an 'universal, everywhere applicable and always delivering the correct estimation' technique does not exist, according to Litke [LI07]. There is also no clear selection progress for the estimation technique to use, beside the time aspect when the use of a technique is available. As shown in figure 2.2, the function point and the COCOMO technique are both possible after the study stage. As a comparative method, the function point technique has not much in common with COCOMO, which is an algorithmic technique. The project manager has to balance the weigh of each technique and probably choose that one he has most experience with. As an example for estimation techniques these will be described here in more detail with a comparison at the end.

### 2.2.1 Function Point

The Function Point technique was first mentioned by Allan J. Albrecht in 1979 at the IBM symposium [AL79]. He declared that an useful measurement of productivity is only possible in relation to the functionality that is visible to the user. This measured productivity needs to be independent of the used technology and is calculated with the proportion of project effort and the allocated function points. This resulted in the idea to turn over this calculation for a preliminary estimation of the effort the project would have. Because of the clarity and the flexibility the technique spread fast.
It helps to estimate the scope of a project to an early stage and is suitable for benchmarking in the own company as well as on national or international level [PO12]. It contains algorithmic and also comparative methods. Basically, this technique uses five steps for estimation [JE01]:

1. Determining the components
2. Evaluation of the components
3. Calculating the function points
4. Categorization of the influence factors
5. Calculating the development effort with the function points and influence factors

The most important part of this technique is that all measurements only include the user view. This means that the users view is focused on that functions that are important for the specific business process. Implemented business processes are the components that have to be determined.

#### Determining the Components

To divide the project in useful components, all inherited business processes are divided into elementary process. These are the smallest and from business perspective view useful and closed activities, that can be performed by the system [PO12]. It is useful to categorize the components, because a change in the datasets

is followed by more effort than changing a request [WI05]. This distinction is made into five categories:

1. Input Data
2. Output Data
3. Request
4. Dataset
5. Reference Data

The *Input Data*, sometimes called *External Inputs* (EI), is an elementary process in which data crosses the boundary from the outside of the application to the inside. This data comes from a data input screen or another application. To maintain one or more logical files, this data can be used for or to control business informations. *External Outputs* (EO), or Output Data, are an elementary process in which derived data passes across the boundary from the inside to the outside. Created output files or reports are sent to other applications. These are created from one or more internal logical files and external interface files.
Internal *Logical Files* (ILF's) or Dataset are an user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through external inputs.
The next category are requests or *External Inquiry* (EQ) or Request. An elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface files. This process does not update any Internal Logical files, and the output side does not contain derived data.
The last category is the *Reference Data* or *External Interface Files* (EIF), which are a user identifiable group of logically related data that is used for reference purposes only. This data resides entirely outside of the application and is maintained by another application [DU14].
The result of the categorization is an amount of components for each category. This categories must then be evaluated.

### Evaluation of the Components

The evaluation of the components is simply the classification of each category in their level of difficulty (simple, medium or complex). After this, each component is multiplied with the point value according to their difficulty.
The following table is described by Wieczorrek and are standard values for the function point technique [DU14]:

### Calculating the amount of Function-Points

After the evaluation there is an amount of components for each of the categories from above. To get the number of function points each category has to be multiplied

| Category | simple | medium | complex |
|---|---|---|---|
| Input Data | 3 | 4 | 6 |
| Output Data | 4 | 5 | 7 |
| Request | 3 | 4 | 6 |
| Dataset | 7 | 10 | 15 |
| Reference Data | 5 | 7 | 10 |

**Table 2.2.1.1.** Point value of each category

according to the selected weigh and all categories are summed. This results in the following equation:

$$E1 = \sum_{1}^{n} (Function \cdot Difficulty) \tag{2.2}$$

*Function* is the respective category and *Difficulty* is the weigh from table 2.2.1.1. The resulted value E1 is necessary for evaluating the estimated points with the influence factor.

### Classification of Influence Factors

Do get a more realistic estimation, all influences that can affect the project surroundings are measured. There are seven defined influence factors [BA08], which are described below.
Each of the influence factor has a value between zero and five which describes how much the factor influences the project.

1. **Integration into other applications**
   The system will work with different applications and will send and receive data from other applications. This states if there is a cooperation with other applications and if the communication exists online or offline.
2. **Local Data Processing**
   This factor describes if the system will work with distributed data. Zero means that the system does not work with other applications and five means that there is an integration into other applications in both ways.
3. **Transaction Rate**
   A high transaction rate affects planing, development, installation and maintenance of the system. It describes how much transactions are to expected with the system.
4. **Processing Logic**
   The processing logic can be divided in 4 subcategories: *Arithmetic Operation*, *Control Procedure*, *Exception Regulation* and *Logic*. Arithmetic Operation describes the intensity of the operations in the project. The controlling of the results is stated within the Control Procedure. The *Exception Regulation* describes how eventual exceptions are treated and the Logic multiplier describes

how much effort is to be expected for planning the logical component of the project.

5. **Reusability**
   How much of the produced software has to be reusable in other projects. A high value means extra effort in the planning stage for module-based development.
6. **Stock Conversion**
   This describes how much of the used data needs to be transformed for the usage within the project. A high value means, that much input data from other applications have to be transformed into data the application can process.
7. **Facilitate Change**
   The application was especially planned and developed in such a way that changes can be made easily. A high value means that the user can make changes on the system on its own and that the changes are available immediately.

When all influence factors are set, all values for each factor will be summed up to the value E2.

$$E2 = \sum_{1}^{n} Influence\ Factor \tag{2.3}$$

This influence factor indicator has then to be transformed to a multiplier that calculable with function points [BA08][DU14].

$$E3 = \frac{E2}{100} + 0,7 \tag{2.4}$$

#### Calculation of the Function Points

With the calculated Function Points (E1) and the Influence Factor multiplier (E3) are now the total-function-points (TFP) can now be calculated with the following formula:

$$TFP = E1 \cdot E3 \tag{2.5}$$

From the regression analysis of previous projects a standard calculation of Function Points per day can be made using table 2.2.1.2.
The project has to be classified with their Total-Function-Points and divided through the appropriate points per day, as described in 2.2.1.2. This results in the expected man days for this project.

### 2.2.2 COCOMO

The Constructive Cost Model (COCOMO) is used when it is not possible to rely on experience. It is based on algorithmic and parametric methods that merge parameters from software projects, combining the system size, product properties, project and team factors with the effort for developing the system [JE01].

| Estimated Size | Function Points | Points per Day |
|---|---|---|
| Small Project | till 350 | 18 |
| Mid Small Project | till 650 | 16 |
| Medium Project | till 1100 | 14 |
| Mid Large Project | till 2000 | 12 |
| Large Project | as of 2000 | 10 |

**Table 2.2.1.2.** Function Points to Days

As an public domain model it is free to use and is considered to be a classic for algorithmic methods. This technique was adjusted to the IT development through the years and is common in many companies for cost estimation. The accuracy of the COCOMO techniques rises with later project stages. The deviation of the cost estimation is at the beginning of the project between *0,25 * MD* and *4 * MD*. In later project stages this variation decrease until it is zero just before the project ending [SO07].
The parameters for COCOMO can be split into three parts. The project classes, model variants and the implementation time and effort.

### Project Classes

The project classes represent the estimated size of the program itself. These are expressed in kilo delivered source instructions (KDSI). COCOMO classifies three project classes with different calculation factors as described in table 2.2.2.1 [SO07].

| Complexity | Calculation Factor | KDSI | Describtion |
|---|---|---|---|
| Small | 1.05 | $< 50$ | A small, well-known project team works together, the environment is well-known, there is no big innovation necessary and no pressure due to a deadline. |
| Medium | 1.12 | 50 - 300 | Employees with average experience, team members with some experiences in subjections are working together. |
| Complex | 1.20 | $> 300$ | High cost and deadline pressure, high innovations and an extensive project. High requirements to the project team and new components. |

**Table 2.2.2.1.** COCOMO project classes

### Model Variants

The COCOMO technique can be differed into basis, intermediate and detailed model. These represent the detail level of the cost estimation.
The first stage is also called basis estimation and is a rough estimation, which estimates the project costs to an early stage of the project. The costs are calculated with an equation without dividing the project into structure or time aspects. The base model is an useful starting point for later estimations.
The second stage adjusts the first estimation to a higher level of detail by differentiating the development stages. It is not a parametric estimation yet, because not all data can be considered.
The detailed model is the last stage of the estimation and allocate the estimation with fifteen influence factors [JE01]. These influence factors are divided into four categories.

1. **Product Attributes**
   a) *Required Software Reliability*
   b) *Size of the Application Database*
   c) *Complexity of the Product*
2. **Hardware Attributes**
   a) *Run-time Performance Constraints*
   b) *Memory Constraints*
   c) *Volatility of the Virtual Machine Environment*
   d) *Required Turnabout Times*
3. **Personal Attributes**
   a) *Influences Analyst Capability*
   b) *Software Engineering Capability*
   c) *Applications Experience*
   d) *Virtual Machine Experience*
   e) *Programming Language Experience*
4. **Project Attributes**
   a) *Use of Software Tool*
   b) *Application of Software Engineering Methods*
   c) *Required Development Schedule*

### Calculation of Implementation Time and Effort

Because of the differences in each model of the COCOMO estimation, each model has its own equation for cost calculation. Each formula calculates the estimated time of the project in person-month (PM), which are stated by Boehm as 152 working hours resulting in one PM, with 19 working days per month and eight hour days [BO81].
The formula for the basic model calculates the complexity of a project with the KDSI value and the calculation factor of the project which gives the calculated

PM as a result. This formula is as follows:

$$PM = Complexity \cdot (KDSI)^{Calculation\ Factor} \tag{2.6}$$

The *KDSI* value is the expected amount of code lines in the project. As described in table 2.2.2.1, the Calculation Factor is derived from the KDSI value. To figure out the complexity multiplicand table **??** describes for each project the suitable solution.
The equation for the intermediate model is the same as for the basic model (2.6) [BO81]. The difference is the changed multiplicand for this model, as described in table 2.2.2.2.
The detailed model is an extension of the intermediate model that adds effort multipliers for each phase of the project to determine the cost drivers impact on each step. The effort is calculated as function of program size and a set of cost drivers given according to each phase of software life cycle [BO81]. For an equation it is necessary to multiply all cost drivers $C_i$ together, as described in the following formula:

$$C = \prod_{i=1}^{k} C_i \tag{2.7}$$

In this formula is $k$ the sum of all influence factors with 15 and each $C_i$ represents one of the 15 influence factors. This results in a combined influence factor which can be multiplied with KDSI value to get the total time for development (TDEV). But therefore the KDSI has to be calculated with the complexity factor for the detailed model from table 2.2.2.2. This calculation can be described in the following equation [BO81]:

$$TDEV = C \cdot (KDSI)^{CalculationFactor} \tag{2.8}$$

| Complexity | Basic Model | Intermediate Model | Detailed Model |
|---|---|---|---|
| Small Project | 2.4 | 3.2 | 0.38 |
| Medium Project | 3.0 | 3.0 | 0.35 |
| Complex Project | 3.6 | 2.8 | 0.43 |

**Table 2.2.2.2.** COCOMO complexity multiplicands

### 2.2.3 Comparison

The Function Point analysis is useful for software projects of all size, mainly desktop based platforms. This is one of the biggest contrapositions as this technique

is not good adaptable for console programs. Instead COCOMO is used for large corporate and government projects, including embedded firmware projects. The major source of criticism at COCOMO estimations is that they are inappropriate for small projects and that it is based on the waterfall model. It is recommended for large and lesser known projects to use COCOMO 2 instead.
The similarities between these models are that they both rely on algorithmic methods, with the difference that COCOMO is a logarithmic type and function point is linear. Both were created in the same time, where the project development cycle relied on linear procedure models and the used programming languages where procedural.
None of the techniques is necessarily better or worse than the other, in fact, their strengths and weaknesses are often complimentary to each other. There is no estimation technique that is the 'best fitting' for a project [AB14].

## 2.3 Software for Cost Estimations - State of the art

The big uncertainty of cost estimations is an actual problem for cost estimations. Stake holders and project managers don't trust the estimation output and calculate a surcharge on all estimations. In big projects this uncertainty factor is from 5% up to 50% is added on the estimated costs for a project [FI10]. That is because many large projects are stopped due to budget overrun or incomplete requirements [ST14], which results in insecurities on the cost estimation output.
Whereas the cost estimation is possible as paper work, there are several software products that allow it.
As there are many start-up IT companies founded in the last five years, there is no start-up company that develops cost estimation software [RI15]. Also, there is no approach in developing a mobile application for cost estimations. Also the use of smartphones increased from 2013 to 2014 by 21% and in the business environment 80% use a smartphone [LO14]. This opens up good opportunities on the market for the application.
There are three cost estimation tools that are mostly referred to.

### 2.3.1 SEER - Cost Estimation Software

The 'Seer - Cost Estimation Software' is developed by Galorath Inc. in Los Angeles, USA. Galorath Inc. offers consulting and software for cost estimation, decision support and project management. The company was founded in 1979 with the goal to improve the software and hardware development process in the industry. The next step was to improve their consulting quality by developing their own tools for this process. *Seer* is the name of their tool set whit a large variety of different tools that support the development process of new products.
Seer for Software is an estimation application for 'estimating, planning, analyzing and managing complex software projects'[1]. Based on the SEER design principles

[1] `galorath.com/products/software/SEER-Software-Cost-Estimation`

the software contains an annotated and guided interface for defining projects, a parametric simulation engine and numerous standard and custom reporting options. Due to an open architecture API the SEER application can be integrated with enterprise applications and departmental productivity solutions. Galorath specifies that all estimations within this software are repeatable and consistent.
According to the software description, 'a high-level software estimate can be developed in a matter of minutes using SEER´s intuitive, window-based interface' [GA11]. A dialog guides the user through the process of creating a new estimation. In the first screen the user has to set project name and decides whether he creates an empty project or starts with a scenario. The scenarios are example projects with some data for starting the estimation. In the next step the user chooses the estimation method he wants to use. The estimation methods are subdivided in functional, lines and sizing scale. Another feature of the software is the documentation and export function. All estimations can be exported to Microsoft Project, Microsoft Office, IBM Rational or other third-party software. SEER delivers a huge variety of estimation methods, guided processes and many possibilities for documentation and reporting of all estimations.
The software can be bought in an estimator, project manager and studio version. The estimator version inherits only a standard estimation. The project manager and studio version allow estimation checking and access to the projects database. The studio version allows also independent crosscheck and verification. The price of each software version is nowhere specified on the homepage and must be requested.

### 2.3.2 PRICE - Cost Estimation Software

PRICE Systems L.L.C. provides agile estimating solutions. With their head office in Maunt Laurel, USA, and 12 locations worldwide they offer since 1969 estimating acquisitions[2]. Their Software Development Cost Model is one of the oldest and most widely used software parametric models for software development projects. In 2003 PRICE released *TruePlanning*, which contains methods that estimate the scope, cost, effort and schedule for software projects.
For cost estimation, purchasing efficiency and budget planning PRICE Systems develops 'multi-faceted cost estimating solutions' [GA11]. Their biggest project is the PRICE Estimating Systems (ESI) Framework that delivers solutions for estimating projects and cost management. This framework is not specialized for estimating only software projects but also other projects.
The homepage describes that it is possible to estimate projects with all current state of the art cost estimations. The collaboration with other users is also possible as well as sharing the results through the program. This allows faster teamwork and a better estimation output. A data driven method for all estimations allows to compare the estimation with already completed projects. *TruePlanning* allows also estimation output to a specific work breakdown structure or cost element structure for accurate top-down and bottom-up estimate comparisons. Mappings can be stored, retrieved, and modified to keep pace even as program activities

[2] `http://www.pricesystems.com/`

change. Beyond specific cost estimating products and services, PRICE Systems offers strategic cost management services. They collaborate with estimators, engineers, project managers, and financial and executive management to design and implement integrated cost management systems that meet the unique challenges in the environment.
Any details about the cost of the software or licensing have to be requested at *Price Systems*.

### 2.3.3 SLIM Estimate

SLIM Estimate is a cost estimating software developed by Quantity Software Management[3] (QSM), an estimation company with their head office in McLean, USA. QSM was founded in 1978 as a software management consultant company for measurement, estimation and controlling projects. Beside their consulting business segment they develop the SLIM Software which contains software for controlling, metrics and estimation.
The SLIM Estimate software provides a flexible cost estimation which align the estimation to the project size. Integrated schedules offer simple exports of an existing calculation. The software can suggest alternate solutions calculated on past projects.
The homepage does not go into detail how this suggestion works. For a new estimation the user can use the SLIM Database and the QSMA database, which can be used for new estimations.
For better use in the company the SLIM Estimate inherits interfaces to MS Office and the possibility export estimations as a web representation. SLIM Estimate delivers hereby a software package with many functionalities. Especially the access to a large database and the knowledge base of QSM is a big plus value to the application. As a result, it is possible to access the experience of past projects and get the benefit from this experience.

### 2.3.4 Commonalities

All described cost estimation tools have in common, that they inherit different estimation techniques. The user can always decide with which technique he wants to use. They are all tools that are on the market for many years and rely on experience from project managers. Each software has a different approach on the implemented estimation process and collaboration of project estimations. They all have a database with previous cost estimations but with extra costs for accessing it.
The User interface is like in many large software projects more functional and does not contain a modern design. None of them has an approach for developing a mobile application and rely on Microsoft Windows as their OS.

[3] `http://qsma.com/slim-estimate`

### 2.3.5 Kinvey

A special software developer is Kinvey from Boston whose main business is the development of mobile applications. They don't develop a certain cost estimation tool, but in terms of general cost estimations, the company is still very interesting[4]. Kinvey offers an online estimation for mobile applications in which the potential customers can estimate the costs for their application. The estimations show the time the customer would have to spend for development and the costs for development at Kinvey. The cost of developing through Kinvey here are always cheaper than a proprietary development.
The cost estimation process is simple. The estimation schedule is a one-page where the user can select the components of his application. The user is guided through some question about features the application should contain. After each step the user can see the actual cost estimation of the project.
The costs are calculated from the selected components in man days. It is not possible to define and add your own components for the estimation. The individual items that the user can choose are sorted by groups and displayed graphically. Estimating the cost for a mobile application development can also be used for own estimations, but there is no way to export this estimation.

[4] `http://www.kinvey.com/mbaas-savings-calculator`

# 3

# Application concept

This chapter describes the project plan and the concepts for the developed application. These concepts are foundation of the developed application and describe how the internal processes in the application work.

## 3.1 Project planning

The project followed the incremental development model. For initial planning the features from existing cost estimation software as described in section 2.3. The next step was creating a *GUI prototype* to show how the estimation would look like on a smartphone. Afterwards each cycle followed the rules of incremental development. An evaluation of the wanted features for the application was done in meetings with my supervisor Prof. Dr. Georg Rock. Those features request were compiled into the project goals and requirements. For the next step they were analyzed and implemented. Testing for the application were made afterwards by some of my fellow students. The result was then presented in a meeting and further features were planned for the next iteration cycle.

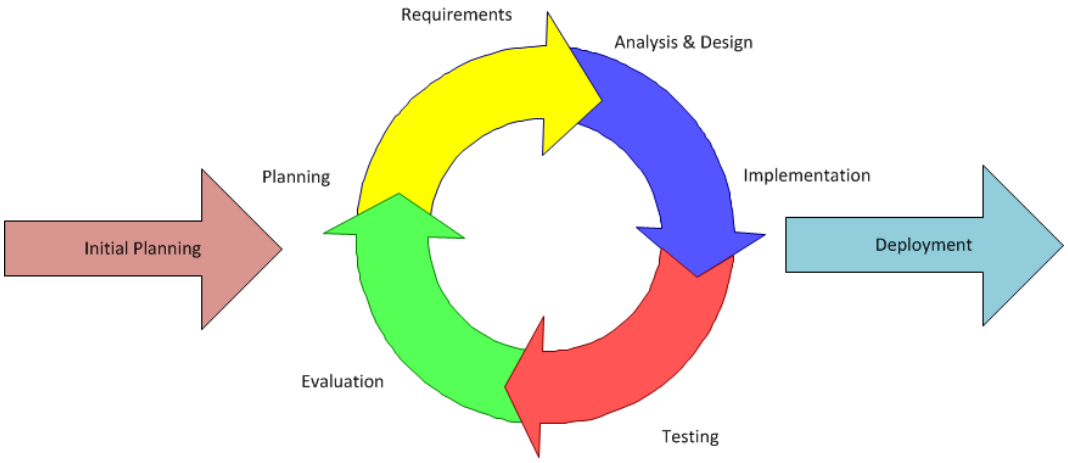


**Fig. 3.1.** - Incremental Development
Source: `https://www.inflectra.com/Methodologies/`

### 3.1.1 Objectives

The project targets are the benefit the application should bring the user. They are linked with the requirements from section 3.1.2 and describe the condition to be achieved with the application. Whereas requirements can change over time, targets stay the same.
Most important target, as described in table 3.1.1.1, is *T01*. 'The application must allow a quick and mobile cost estimation of IT Projects' describes that the cost estimation has to be implemented for mobile device use. Also, the cost estimation has to be fast and can use features of the mobile platform to achieve this target.
As the developed application is only a beta version, it has to be developed modular for future feature implementation, as described in *T04*.
*T06* describes the target that represents an admired feature. The application should provide the possibility to compare projects and measure their relation. This allows to check previous projects for their cost estimation, that the user has a reference how much effort similar projects took.
The target for this bachelor thesis is to implement the function point estimation technique. This is the showcase for this application and shows the possibility of cost estimations for software project on mobile devices and is described in *T09*.
All remaining targets are medium priority and describe what processes to achieve with this project.

| ID | Target |
|---|---|
| T01 | The application must allow a quick and mobile cost estimation of IT Projects. |
| T02 | The application must provide informations how cost estimations work. |
| T03 | The application have to improve the cost estimation during the project life cycle. |
| T04 | The application architecture has to be modular to add new cost estimation methods and analysis tools faster and more efficient. |
| T05 | With the application it is possible to get the information of completed projects and take advantage of their estimation. |
| T06 | The application allows comparison between projects and shows projects that are related to each other. |
| T07 | The application gives information about the estimated man days of a project and the estimated costs. |
| T08 | A project analysis within the application allows an overview of the project changes and gives detailed information about the estimation. |
| T09 | Till the end of the bachelor thesis the function point method has to be implemented and be fully operational. |

**Table 3.1.1.1.** Project Targets

### 3.1.2 Requirements

All requirements for the implemented application are based on the targets described in section 3.1.1 and from the regular meetings. All requirements are described in the appendix XXXX and meet the demands for requirements as described by Balzert [BA09].

### 3.1.3 Cost Estimation

For the evaluation of the needed time, the project was estimated with the function point technique. Therefor all functionalities are grouped into functional groups, as described by Poensgen [PO12]. Afterwards each function was assigned a category for the estimation, as described in 2.2.1.
The group Project, as described in 3.2, inherits all functions for a project. All function groups for the application can be found in the application. The project functionalities in 3.2 are created with the requirements. Some requirements can be grouped into one functionality. As an example the function *Create/Edit Project* inherits all requirements described in the appendix XXX. This function is marked as an *EI* and *ILF* because all changes are made by the user for this function and creates or edit all internal files for the project.

Functions that expects an input from the user are marked as *EI*, like delete project. Output functions like show estimated cost are a *EO* function, because they only display values and results to the user. Functions with a calculation are described as *EQ* combined with *EO* if they produce an output like *Export Project* or process an *Input* like *Change Estimation*.The total functional group can be found in the appendix. This categorization of the function groups was inserted into the estimation table as described in table 3.1.3.1.Summed up the project has total **246** *Function Points*.

| Category | Amount | Classification | Weight | Sum of Row |
|---|---|---|---|---|
| Input Data | 17 | medium | 4 | 68 |
| Request Data | 6 | simple | 3 | 18 |
| Output Data | 8 | medium | 5 | 40 |
| Dataset | 8 | complex | 15 | 120 |
| Reference Data | 0 | simple | 5 | 0 |
| **Sum** | | | | **246** |

**Table 3.1.3.1.** Estimation - Total Points

As described in section 2.2.1, the next step is to set the influence factors which are described in table 3.1.3.2. The influence factor *Integration into other applications* is set to two because the application has no direct communication to other software. The application does not work with other applications but has local data,

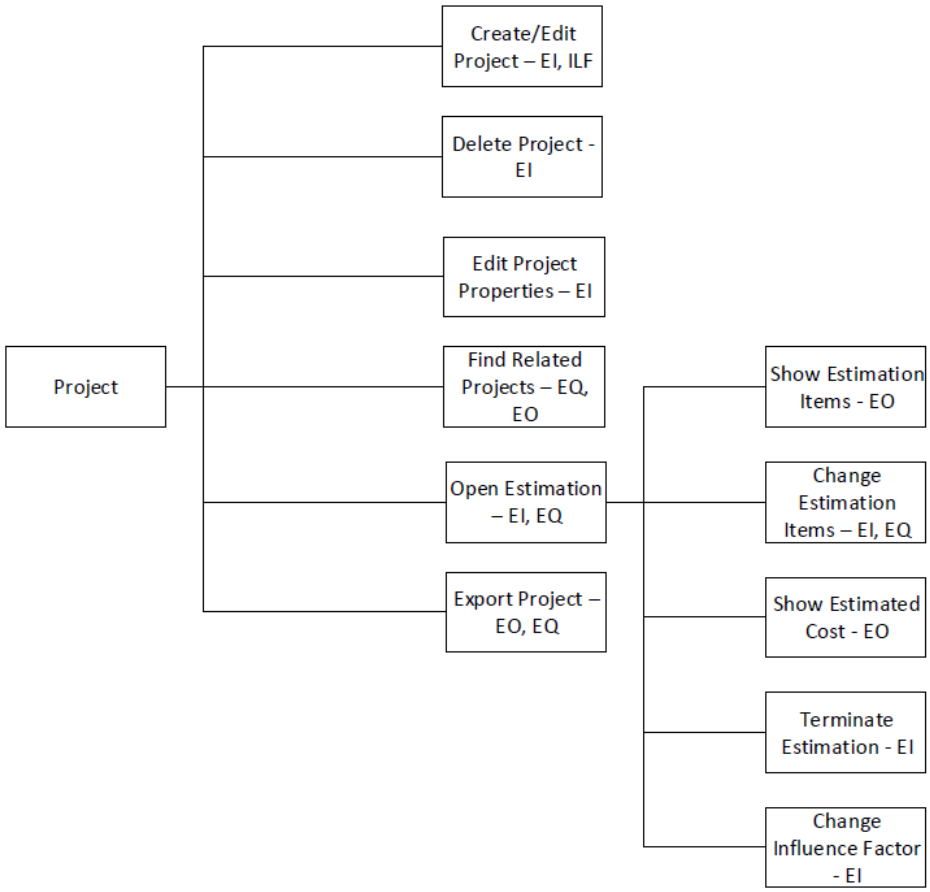


**Fig. 3.2.** - Function Group - Project

like the databases that have to be processed which is the reason for setting the factor *Local Data Processing* to two. As the application is only a local software for the end users device their no much transaction to be excepted. To assure that there is enough time planned to implement a access to the database this factor is set to one. There are many algorithms and equations in the application which have to be checked for errors and exceptions. Also a high effort for the logical component is to be expected. Because of this all influence factors in the *Processing Logic* group are set to the second highest value. For further development of the application a high effort has to be expected for *Reusability* in the project which is the reason for this influence factor to be set at two. There is not much data to be expected from other applications. Only the database cause a higher effort with queries to transform the data from the database in readable informations for the application. Changes in the application can not be made by the user. He can change settings and some appearances which is why this influence factor is set to one. Completed with the equation from 2.2.1 all influence factors result in a multiplier of **1.01**. Together with the total function points, the influence multiplier is calculated with the equation 2.5 which give **243.54** points as result. Calculated with the points per

day from table 2.2.1.2 the project is estimated to take **14** man days. In section 6.1 it is analyzed how this estimation has worked out and how much time the project took.

| Influence Factor | Weight |
|---|---|
| Integration into other applications | 2 |
| Local Data Processing | 2 |
| Transaction Rate | 1 |
| Processing Logic | |
| - Arithmetic Operation | 4 |
| - Control Procedure | 4 |
| - Exception Regulation | 8 |
| - Logic | 4 |
| Reusability | 3 |
| Stock Conversion | 2 |
| Facilitate Change | 1 |
| **Sum** | **31** |

**Table 3.1.3.2.** Estimation - Influence Factors

## 3.2 Architecture

The architecture of the project is described as a component diagram in fig. 3.3. As usual in Android developments all resources are stored in the *resources* folder and are accessible from all other classes. The resources inherits *strings*, *images* and many more. Most used in the same folder is the *Layout Data* which inherits all informations for displaying the user interface. Layout informations are connected with the *Activities* and *Fragments* component which are responsible to display the user interface.
The *Project Analyzer* component is responsible for collecting the data from all projects and put them together for analysis. As the name says, *Project Data* contains all informations for a single project. For reading and storing data to the database the component *Database Helper* is the part that allows simple *Data Requests* and *Data Insertions* to the database. It also allows a connection to the *XML Helper* which reads data that is stored to XML files. The

## 3.3 Components

This section describes the most important components from figure 3.3 in detail and what concepts they inherit.

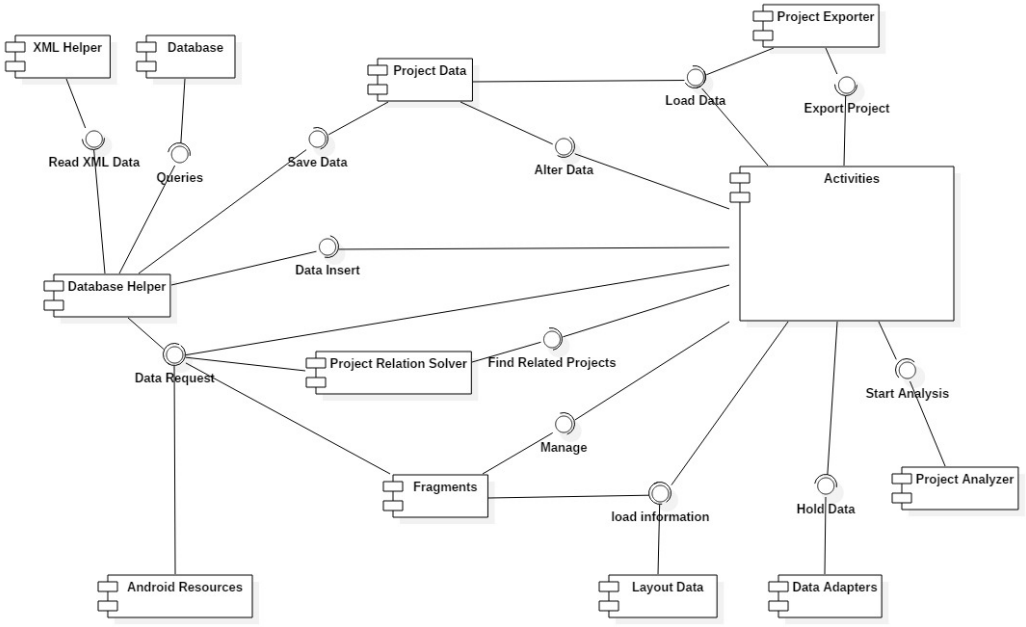


**Fig. 3.3.** - Component Diagram

### 3.3.1 Database Helper

The *Database Helper* component is responsible for accessing the database, where all project informations are stored. Every Class that needs access to the database should only create an object of this component and can read or write data to the database.
The component should contain name and path to the database and allow the initialization of the database. As described in fig. 3.4, with the message *Open Constructor*, an initialization of the database should be done in the constructor of classes that need access to it. This will opens the *SQLite* database and ensures that request on the database can be performed and no error occurs while accessing a non initialized database. All functionalities that are working on the database are in fig. 3.4 combined to the message *Using Class* and combines the three categories for working with the database *SELECT*, *ALTER* and *DELETE*.
For requesting data from the database a Java class has to send a request with the table name and the selection criteria to the Database Helper. This will be transformed into a query which is send to the database. Before sending this data back to the calling Java class the database Helper has to transform the *SQL* query result into readable information. Editing existing informations in the database is done with the message *Alter Existing Data* and the parameters table name and data to be edited. Before sending data to the database the *Database Helper* has to prepare it to insert them in the right column of the table. This query will return true if inserting was successful or false if it was not. Last message option is *Delete Data* which will delete entries from the database. The parameter *tablename* and *Data ID* are the identifiers for a selected entry. Before deleting data from the

database the *Database Helper* has to check if there are any other tables depending to the selected entry and have to ensure that they are deleted too if necessary. After all depending tables are identified the Database Helper will send the delete query to the database. It will then return a boolean value for the deletion process which is forwarded to the Java Class.

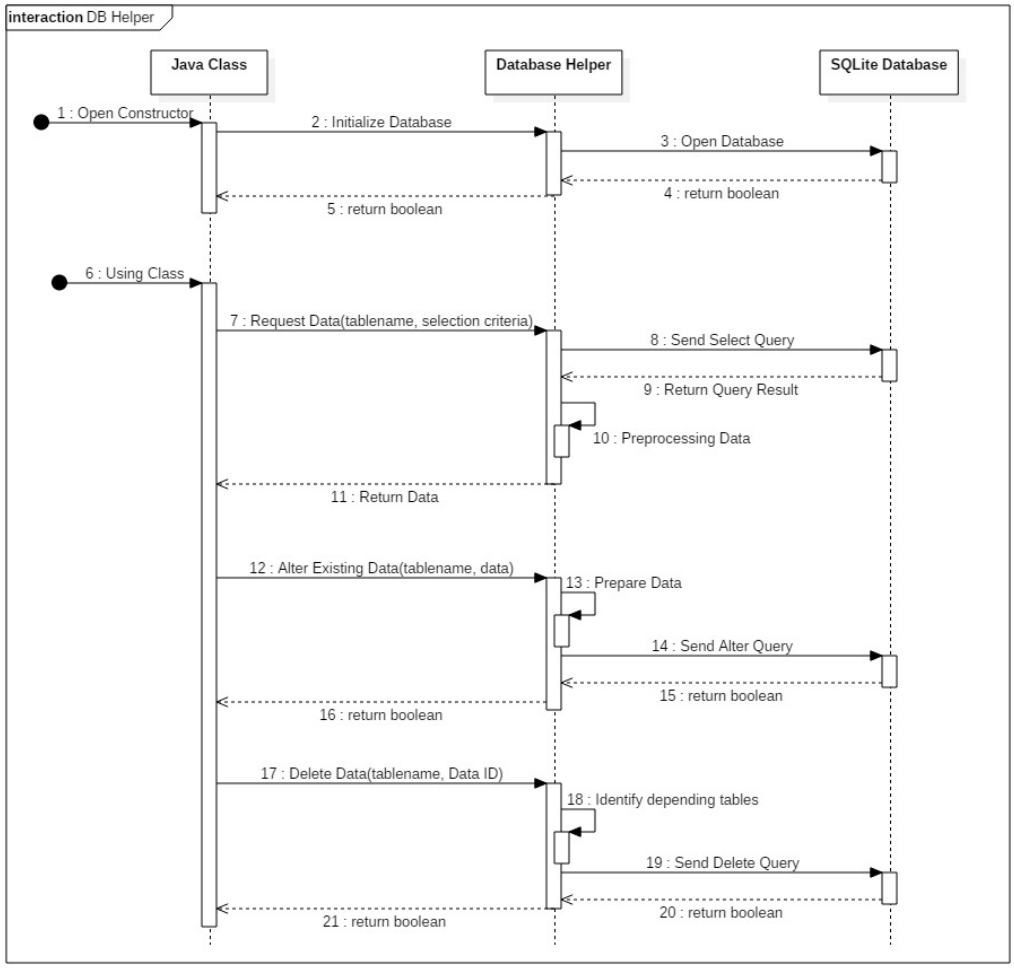


**Fig. 3.4.** - DB Helper Sequence Model

### 3.3.2 Project Data

The *Project Data* component was created to make combine all data of a project from the database and make it accessible for other classes. It contains *Name*, *Description*, *Creation Date* and *Estimation Technique* at the first level of the component. This makes it possible to get a fast overview of the different projects in the application. For estimation specific purposes the *Estimation Items* and *Influence Factor* are minor components for the project data and differ from the projects estimation technique. The *Properties* are also a minor component to build a module-

based structure. As the output information of estimation techniques is the same, the main component contains the amount of *Estimated Man Days* for the project. If the project is completed the *Final Man Days* represent the time the project really took. This value is important to improve future estimations as described in section 3.6, where the improvement process is specified.

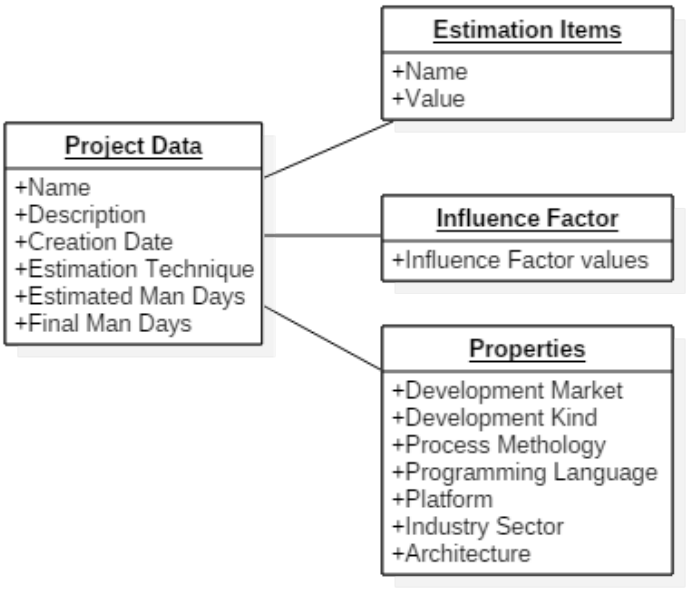


**Fig. 3.5.** - Object Diagram for Projects

### Project Properties

This minor component contains the information that is needed for the *Project Relation Solver* in section 3.3.3. It is a data component which stores all properties that allows reading and editing. Access to is granted to the properties only through the project component which assures that a property is bound to a project.
The available properties are based upon the parts a project can be split to. Not every available factor of a project makes sense for categorizing it. For example a categorization into the size of the project makes no sense as this cannot be compared to a project that is not estimated or developed. Although a categorization in the costs makes no sense as it cannot be compared to a project that has to be estimated. For this reason the available properties are *Development Market*, *Development Type*, *Procedure Model*, *Platform*, *Industry Sector*, *Programming Language* and *Software Architecture*. These values categorize a project without saying something about the size or cost and make it possible to compare them.
All project properties contain only a selection of the possible values as a showcase. For each category are further options possible. For properties where this is a im-

portant factor these are combined in the option *other*.

#### Estimation Items

All items for the estimation are in the *Estimation Items* components. These are the items for an estimation technique that are responsible for the estimation itself. In the *Function Point* estimation for example, these items calculate the value *E1* from the equation 2.2.

#### Influence Factors

This component contains all informations for *Influence Factors*. These are *name*, *chosen value* and the *limit* for the value. This component is also responsible for preprocessing the influence factor for usable information at the calculation of the estimation. In the Function Point estimation for example, this component calculated the value *E2* and *E3* that are described in section 2.2.1.

### 3.3.3 Project Relation Solver

To fulfill the target *T06* from section 3.1.1, this component is responsible to find related project. The point of this component is to compare already existing projects with a selected one. This should help the user to get an overview of how similar projects were estimated. Furthermore the *Project Relation Solver* should allow to transfer the estimation from a related project to the selected to have an estimation where the user can build on.
The *Project Relation Solver* compares therefor an selected project with every other available project from the database. This comparison is made in mainly three steps.

#### *1. Determination of the Property Distances*

Each project has seven different properties which are crucial for finding a relation plus the estimation technique. These properties are *Development Market*, *Development Kind*, *Procedure Model*, *Platform*, *Industry Sector*, *Programming Language* and *Software Architecture*. The first step for calculating the project relation is to compare each property and get the distance. Each of them has its own calculation for the distance and a different weight to fit in the *total distance* equation.

#### *Development Market*

The *Development Market* property stands for the target market of a project and has three possible options: *Inhouse Development*, *Customer Development* and the *Anonymous Market*. For calculating the distance between development markets it was analyzed for whom the project is developed, what is the main factor for the budget, how high is the risk of development and how predictable profit is.

The *Inhouse Development* is a for developing projects in the company. They are usually have the same procedure as other project despite the fact that they don't yield a profit and have a low risk of development and a fixed budget. In the *Customer Development* the risk depends on the customer needs but are predictable. Profit is to be expected high and is connected with the project budget. By *Anonymous Market* developments a high risk is to be expected. Regardless of market analysis it is unsure that the project will be a success and generate profit. The budget for this market type is set from the company itself.
All distances between these markets are described in table 3.3.3.1 and are based on the above description. For example, the development for a customer is quite similar to an anonymous market development the distance between those is set to one. The big difference is that the profit is unsure for the anonymous market development. When comparing similar markets to each other the distance is zero as they are the same.

| Development Market | Compared to | Distance |
|---|---|---|
| Inhouse Development | Customer Development | 2 |
| Customer Development | Anonymous Market | 1 |
| Anonymous Market | Inhouse Development | 3 |

**Table 3.3.3.1.** Development Market - Distance

#### *Development Type*

Different *Development Types* describe the main reason for the project. A *New Project* is developed if there is no previous project to build on or is a new conceptual formulation. An *Extension Project* is made for expansion of existing software from previous projects. The third type are *Research Projects* which are usually made to test an idea or create a prototype for an analysis. These development types can be differentiated in the *main target* of the project, *customer* and *profit level*. To introduce this differentiation the profit level describes the expected money and is divided into values from zero to two. *Zero* declines that the project does not generate any profit. The value *one* expects medium profit and *two* expects a high profit for the project.
The distances described in table 3.3.3.2 are based on the above descriptions. A comparison between new projects and research project has the distance two. A research project does not generate any profit and has a different target than a new project. The same applies with the comparison of an extension project with a research project. By comparing a new project to an extension project the distance is set to one because the expected profit is smaller than in a new project. The reason for this is that an extension project builds up on a previous project and is an addition to a previous contract which normally get a smaller budget.

| Development Type | Compared to | Distance |
|---|---|---|
| New Project | Research Project | 2 |
| Extension Project | New Project | 1 |
| Research Project | Extension Project | 2 |

**Table 3.3.3.2.** Development Market - Distance

### *Procedure Model*

The next property is the used procedure model. There are six procedure models for selection available. The distance in table 3.3.3.3 is calculated by differentiating each procedure by model type, suitable team size, formalization, industry focus and process cover. Model type simply splits the procedure models in sequential and iterative types. The suitable team size indicates the laid out size of team members are useful for this model. Formalization indicates how much has to be put into the documentation of the process steps and the process cover describes if the procedure model involves all main project processes which are development, test project management, quality assurance and the change management.

| Procedure Model | Compared to | Distance |
|---|---|---|
| Scrum | V-Model | 5 |
| Waterfall Model | V-Model | 3 |
| Spiral Model | V-Model | 4 |
| Iterative Development | V-Model | 4 |
| Prototyping | V-Model | 6 |

**Table 3.3.3.3.** Procedure Model - Distance

### *Platform*

Platform is the property that describes where the implemented software is deployed. The difference between platforms is calculated with the porting costs from one platform to another. This costs are calculated with the main programming language of the platform, quality assurance, system type and the deployment difficulty.

| Platform | Porting to | Costs |
|---|---|---|
| Android | IOS | 3 |
| Android | Windows Phone | 3 |
| Android | Mac OS | 5 |
| Android | Linux | 4 |
| Android | Windows 7 | 5 |
| Android | Windows 8 | 5 |
| Android | Windows 10 | 5 |
| Android | Windows Universal App | 6 |
| Android | Web Development | 5 |
| Android | Other | 7 |

**Table 3.3.3.4.** Platform - Porting cost

#### *Industry Sector*

For the industry sector there is no calculated distance as it is mostly an empiric property. It simply compared if the selected sectors are equal or not. If the industry sector does not apply to one of the available sectors there is the option to select *Other* as sector. For the calculation the distance between industry sectors is set to *zero* if they are equal and *one* otherwise. The available options for this property are:

- *Agriculture*
- *Automotive*
- *Banking*
- *Bars and Restaurants*
- *Business Service*
- *Construction*
- *Education*
- *Electronics*
- *Entertainment*
- *Finance*
- *Health*
- *Internet*
- *Music Production*

#### *Programming Language*

The Programming Language describes the main language for the project. As an example for the platform Android the main programming language would be Java. This Property has the most values for calculating the distance between two options. It is divided in two parts. Part one is calculating distance on the basis of programming language properties as described by Ruknet [RU95]. He categorized the programming languages with 12 different properties:

1. **Imperative**
   This value focuses in describing how a program operates. It simply exists on a sequence of instruction. This value is distinguished with *true* or *false*.
2. **Object Oriented**
   This Property describes if the language supports the concept of objects.
3. **Functional**
   These property is a programming paradigm that treats computation as the evaluation of mathematical functions.
4. **Procedural**
   This programming paradigm allows routines and subroutines and contains a series of computational steps.
5. **Generic**
   Generic programming is a style which allows algorithms to be written in terms of types *to-be-specified-later* that are then *instantiated* when they are needed.
6. **Reflective**
   This is the ability of a programming language to examine and modify its own structure and behavior.
7. **Event-Driven**
   An event-driven programming language is determined by events such as user actions or messages from other programs.
8. **Failsafe**
   The ability to throw exceptions if an operation or other system calls fails is known as fail safely.
9. **Type Safety**
   States if a programming language discourages or prevents type errors which are stated in the values *safe* or *unsafe*
10. **Type Expression**
    This property describes if the programmer must explicitly associate each variable with a particular type.
11. **Type Compatibility**
    This implies the use of a type checker in the programming language which verifies if an expression is consistent with the expected type by the context in which that expression appears.
12. **Type Checking**
    A process of verifying and enforcing the constraints of types.

Each property is converted to a numerical representation form which represents each of the possible values. This allows a calculation of the *Distance* of two programming languages. As described in equation 3.1 this calculated with the absolute value of each category. The constant $c$ is the amount of all categories, which is twelve. For each category $K_i$ of the compared programming language the value of the other language is subtracted. This summed up is the distance of these two languages.

$$Distance = \sum_{i=1}^{c} |K_i - K_i'| \tag{3.1}$$

| Programming Language | imperative | object oriented | functional | procedural | generic | reflective | event driven | failsafe | type safety | type expression | type completion | type checking |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C++ | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 2 | 1 | 0 |
| Java | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | 1 | 2 | 1 | 1 |

**Table 3.3.3.5.** Programming Language - Distance

The second part is the conversion cost in another language which is the estimated effort for converting the source code. For calculating these *Conversion Costs* the equation 3.2 delivers an estimated cost factor. These costs are simply the *Amount* of different categories plus an constant effort factor of one.

$$Conversion\ Costs = Amount + 1 \tag{3.2}$$

The available languages are C, C#, C++, COBOL, Clojure, Java, Javascript, Matlab, Objective-C, PHP, Prolog, Python and Scala. The complete categorization and converting costs of all programming languages can be found in the appendix XXX.

| Programming Language | Converting to | Distance |
|---|---|---|
| Java | Cpp | 4 |

**Table 3.3.3.6.** Programming Language - Conversion Cost

#### *Software Architecture*

asd

#### *2. Determination of the Total Distance*

asdasdasd

#### *3. Calculation of the Relation*

asdasdasd
asdasd

| **Software Architecture** | Category | overall agility | ease of deployment | testability | performance | scalability | ease of development |
|---|---|---|---|---|---|---|---|
| Client-Server | interactive | 2 | 1 | 3 | 3 | 3 | 2 |
| MVC | distributed | 2 | 3 | 2 | 3 | 3 | 2 |

**Table 3.3.3.7.** Software Architecture - Distance

### 3.3.4 Project Analyzer

Aufbau der Analyze der Projekte, Wie die Graphen vorgestellt. Sinn davon

### 3.3.5 Minor Components

Export, statistic, Help DB, Feedback, Project Filter, accessing resources

## 3.4 Database design

Vorheriges Design der zentralen Datenbank. Wichtig um die Klassen anzupassen und vorher schon mit wichtigen Daten zu füllen

### 3.4.1 Project Database

Datenbank für alle Projekte, Eigenschaften, Einflussfaktoren

#### Project Properties

Welche Tabellen gibt es. Wichtige Tabellen, Aufbau der Tabellen und Grund

#### Influence Factors

Wie die Einflussfaktoren Aufgebaut, was ist der Gedanke dazu?

#### Projects

Wie sind Projekte gespeichert, Aufteilung in Projekt, Projektdetails und zugehörige Tabellen, wie die Schätzung Organisiert und wie der Zugriff auf die Elemente

### 3.4.2 Userinformation Database

Datenbank für Spätere Synchronisation und Userinformationen vom Server, Konzept dazu

## 3.5 User Interface

### 3.5.1 Projects Overview

Anordnung der Projekte, Wichtige Informationen zum sehen, Filtern und Suchen nach Projekten

### 3.5.2 Project Creation

Komponente zur korrekten Erstellung von Projekten, Geführte Eingabe, Korrekte Erstellung von Projekten in der DB, Swipe Funktion

### 3.5.3 Estimate Function Point Project

Aufbau der Schätzung, Umwandlung von Tabelle in App

### 3.5.4 Influence Factors

Aufbau der EInflussfaktoren, Neue anlengen

### 3.5.5 Analysis

Wie die Analyse aufgebau und was soll diese bringen?

## 3.6 Adjusted Estimation Process

### 3.6.1 Continuous Improvement of the Estimation

# 4

# Implementation

Was zur Implementierung genutzt + Used Libraries

## 4.1 Project Structure

Aufbau des Android Projektes und Strukturierung

## 4.2 Use of an exisiting Database in Android

Wie und warum existierende Datenbank in das Projekt eingebunden

## 4.3 The Database Helper

Größe der Klasse und Grund dafür, Ein Codefragment besprechen

## 4.4 Using multilingual Strings with the Databse and Resource Files

Hintergrundgedanke, Wie Funktioniert es, Beispiel Aufruf

## 4.5 Calculate Person Days

Wie läuft die Kalkulation ab

## 4.6 Find related Projects

Wie ist der Algorithmus implementiert

## 4.7 ListView Elements and ViewHolder

Schwierigkeit bei Listviews, Lösung mittels dem ViewHolder besprechen

# 5

# Software Test

Ablauf/Aufbau der Tests

## 5.1 Test Cases

Beschreibung eines Testfalls, anzahl testfälle, wozu die Testfälle

## 5.2 User Review

Meinungen von Testern, Verbesserungspotential

# 6

# Conclusion

sdfsdsdff

## 6.1 Results

Es ist möglich dadurch mobile Schätzungen zu machen, stabile App und verbessert stetig die Schätzungen neuer Projekte, Ohne Vorkenntnisse ist es schwierig die Anwendung zu Benutzen da der Nutzer nicht weiß wie die Schätzungen funktionieren

## 6.2 Retrospection

Probleme die Schätzung Nutzerfreundlich zu gestalten, Problematik mit dem Projektvergleich war ein schwieriges Problem

## 6.3 Outlook

Implementierung weiterer Schätzmethoden, weitere Features, Anbindung an einen Server, Anpassung auf Tablets, WebTool

# References


ISBSG10. ISBSG : *Practical Software Project Estimation: A Toolkit for Estimating Software Development and Duration.* Mcgraw-Hill Education - Europe, 2010

HU11. Hummel, Oliver : *Aufwandsschätzungen in der Software- und Systementwicklung kompakt.* Spektrum Akademischer Verlag, 2011.

PO12. Poensgen, Benjamin: *Function-Point-Analyse: Ein Praxishandbuch* . dpunkt.verlag, 2012.

GE11. Geirhos, Matthias: *IT-Projektmanagement: Was wirklich funktioniert - und was nicht* . Galileo Computing, 2011.

MC06. McConnell, Steve : *Software Estimation: Demystifying the Black Art: The Black Art Demystified).* Microsoft Press, 2006.

BA08. Balzert, Helmut : *Lehrbuch der Softwaretechnik: Softwaremanagement.* Spektrum Akademischer Verlag, 2008.

BA09. Balzert, Helmut : *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering.* Spektrum Akademischer Verlag, 2008.

HU04. Hürten, Robert: *Function-Point Analysis - Theorie und Praxis: Die Grundlage für das moderne Softwaremanagement.* expert, 2004.

CA04. Camarinha-Matos, Luis M.: *Evaluating a Software Costing Method Based on Software Features and Case Based Reasoning.* expert, 2004.

ME16. Meyer, Dagmar : *Projektmanagement - Aufwandsschätzung.* Ostfalia - Hochschule für angewandte Wissenschaften. *(Stand: 04.02.2016)*

WE16. Weigand, Florian: *Aufwandsschsatzung in IT-Grosprojekten - Function Point Methode.* TU München. *(Stand: 04.02.2016)*

FI16. Fischmann, Lee : *Evolving Function Points.* Galorath Inc. *(Stand: 04.02.2016)*

OM14. OMG: *Automated Function Points (AFP).* OMG - Object Management Group *(Stand: 04.02.2016)*

DU14. Dumslaff Uwe, et al: *Studie IT-Trends 2014.* Capgemini Deutschland Holding GmbH
`https://www.de.capgemini.com/resource-file-access/resource/pdf/capgemini-it-trends-studie-2014.pdf`. *(Stand: 04.02.2016)*

PA10. Partsch Melmuth: *Requirements-Engineering systematisch.* examen.press, 2010.

DU14. Longstreet David: *Fundamentals of Function Point Analysis* . softwaremetrics `http://www.softwaremetrics.com/files/Fundamentals%20of%20Function%20Point%20Analysis.pdf`. *(Stand: 04.02.2016)*

WI05. Wieczorrek,Hans W: *Management von IT-Projekten. Von der Planung zur Realisierung.* expert, 2005.

JE01. Jenny, Bruno: *Projektmanagement in der Wirtschaftsinformatik.* vdf Hochschulverlag AG an der ETH Zürich, 2001.

AL79. Albrecht Allan J.: *Measuring Application Development Productivity.* IBM Corporation, 1979.

LI07. Litke, Hans-Dieter: *Projektmanagement: Methoden, Techniken, Verhaltensweisen.* Carl Hanser Verlag GmbH & Co. KG, 2007.

WI16. WinfWiki: *Methoden und Verfahren der Aufwandschätzung im Vergleich* . Intermoves GmbH *(Stand: 04.02.2016)*

GO16. Google: *Android Design Principles* . Google Inc. `http://developer.android.com/design/get-started/principles.html`. *(Stand: 04.02.2016)*

KA16. Kathleen, Peters: *Software Project Estimation.* i.c. stars, 2016.

SO07. Sommerville, Ian: *Software Engineering.* Pearson, 2007.

BO81. Boehm, Barry: *Software Engineering Economics.* Englewood Cliffs, 1981.

AB14. Abubaker Ali ,Noraini Ibrahim : *Comparative Analysis Between Fpa and Cocomo Techniques For Software Cost Estimation.* International Journal of Engineering Research and Development, 2014.

FI10. Fischer, Christian, Aier, Stephan: *IT-Projektkostenschätzung – Ein pragmatischer Ansatz.* Institut für Wirtschaftsinformatik, Universität St. Gallen, 2010.

ST14. The Standish Group, Aier, Stephan: *The Standish Group Report.* Project Smart, 2014.

RI15. Ripsas, Sven , Tröger, Steffen : *3. Deutscher Startup Monitor* . KPMG, 2015. `http://deutscherstartupmonitor.de/fileadmin/dsm/dsm-15/studie_dsm_2015.pdf` *(Stand: 04.02.2016)*

LO14. Lopez, Carola : *Faszination Mobile - Verbreitung, Nutzungsmuster und Trends.* BVDW , 2014. `http://www.bvdw.org/presseserver/studie_faszination_mobile/BVDW_Faszination_Mobile_2014.pdf` *(Stand: 04.02.2016)*

PR15. PRICE Systems : *PRICE Systems - Overview.* PRICE Systems , 2015. `http://www.pricesystems.com/en-us/leadership/priceoverview.aspx` *(Stand: 04.02.2016)*

GA11. Galorath Incorporated : *SEER for Software.* Galorath Incorporated , 2011.

http://galorath.com/wp-content/uploads/2014/08/SEERforSoftware2.pdf *(Stand: 04.02.2016)*

RU95. Ruknet, Cezzar: *A Guide to Programming Languages: Overview and Comparison*. Artech House Publishers, 1995.

# A

## Erklärung der Kandidatin / des Kandidaten

☐ Die Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen- und Hilfsmittel verwendet.

☐ Die Arbeit wurde als Gruppenarbeit angefertigt.

| | |
|---|---|
| Datum | Unterschrift der Kandidatin / des Kandidaten |